Relative Monads and Relative Adjunctions

James Chapman
Institute of Cybernetics, Tallinn

(joint work with Thorsten Altenkirch and Tarmo Uustalu)

This talk does not contain Kan extensions

Plan

- Ordinary adjunctions
- Ordinary monads
- Ordinary monad examples
- Relative adjunctions
- Relative monads
- Relative monad examples

Background/Motivation

- Monads are a very successful abstraction in functional programming
- Mathematics is strewn with adjunctions
- Monads and (the intimately related) adjunctions are perhaps the central topic in (basic) category theory
- But! Some persuasive examples don't quite fit:
  - Monad-like but not an endofunctor
    - often an embedding is involved
  - Satisfy the monad laws but not the structure!

All you need to know about category theory in one slide

- A category is like a set: it has elements called (objects) but also morphisms/arrows between them
- A functor is a morphism between categories
- A natural transformation is a morphism between functors

Adjunctions

$$\begin{tikzcd} \mathbf{C} \arrow[d, bend left=60, "G"] \\ \mathbf{D} \arrow[u, bend left=60, "F"] \arrow[u, phantom, "\dashv"] \end{tikzcd}$$

Trivality ↑

Equality	$\mathbf{C} = \mathbf{D}$ $F = G = Id$
Isomorphism	$Id = G . F$ $F . G = Id$
Adjunction	$\eta : Id \Rightarrow G . F$ $\varepsilon : F . G \Rightarrow Id$

Weakness ↓

Adjunctions can be given by F, G with η and ε or:

F, G, and a bijective map:

$\Phi : (F\ X \rightarrow Y) \leftrightarrow (X \rightarrow G\ Y)$ which is natural in X and Y

Adjunctions

"Adjoint functors arise everywhere"
- Saunders Mac Lane

- An example from logic/functional programming:
  - $\wedge$ is left adjoint to $\Rightarrow$
  - put another way:

$$\begin{tikzcd}[row sep=small] (A \times B) \to C \arrow[d, bend left=60, "\text{curry}"] \\ \hline\hline A \to (B \to C) \arrow[u, bend left=60, "\text{uncurry}"] \end{tikzcd}$$

  - so $(- \times B)$ is left adjoint to $(B \to -)$

Monads from Adjunctions

$$\begin{tikzcd}
\mathbf{C} \arrow[d, bend left, "G"] \arrow[d, phantom, "\dashv"] \\
\mathbf{D} \arrow[u, bend left, "F"] \arrow[loop, out=200, in=-20, distance=12em, "T"']
\end{tikzcd}$$

$T = G \,.\, F$
and the monad operations are derived from the operations of the adjunction

The inverse is also true: every monad can be split into an adjunction in several canonical ways (Kleisli and EM).

Monads

- Monads are given by the following data:
  - $T : \mathbf{C} \rightarrow \mathbf{C}$
  - $\eta : X \rightarrow T\,X$
  - $(-)^* : (X \rightarrow T\,Y) \rightarrow (T\,X \rightarrow T\,Y)$
- Every adjunction gives rise to a monad where
  - $T = G\,.\,F$
  - $\eta = \eta$
  - $(-)^* = G\,.\,\Phi^{-1}$

Ordinary monad example

```
data Maybe A : Set where
  Just    : A → Maybe A
  Nothing : Maybe A

T = Maybe
η = λ a → Just a
(-)* : (A → Maybe A) → (Maybe A → Maybe B)
(-)* = λ k x → case x of
   Nothing → Nothing
   Just x  → k x
```

example (cont.)

```
-- type inferrer (might fail)
infer : Program → Maybe Type
infer p = ...

-- other function (uses monadic interface)
other : {M : Monad X} ... → M Nat
other ... = do
  ...
  type ← infer p
  ...
```

later we might improve infer by producing error messages etc. Don’t need to rewrite other.

Relative Adjunctions

$$\begin{tikzcd} & \mathbf{D} \arrow[dr, bend left, "G"] & \\ \mathbf{J} \arrow[ur, bend left, "F"] \arrow[rr, bend right, "J"'] & \dashv & \mathbf{C} \end{tikzcd}$$

Relative adjunctions are given by F, G, J and a bijective map:

$\Phi : (F\ X \rightarrow Y) \leftrightarrow (J\ X \rightarrow G\ Y)$ which is natural in X and Y

We can have a unit ($\eta : J \Rightarrow G\ .\ F$) but no counit

Relative Monads from Relative Adjunctions

$$\begin{tikzcd}
& \mathbf{D} \arrow[dr, bend left, "G"] & \\
\mathbf{J} \arrow[ur, bend left, "F"] \arrow[rr, bend right, "J"'] \arrow[rr, bend left=90, looseness=2, "T"] & \dashv & \mathbf{C}
\end{tikzcd}$$

As before, $T = G \,.\, F$ and the (relative) monad operations are derived from the operations of the (relative) adjunction

As before, every (relative) monad can be split into an (relative) adjunction in several canonical ways (rel. Kleisli and rel. EM).

Relative Monads

- Relative Monads are given by the following data:
  - $T : \mathbf{J} \to \mathbf{C}$
  - $J : \mathbf{J} \to \mathbf{C}$
  - $\eta : J\,X \to T\,X$
  - $(-)^* : (J\,X \to T\,Y) \to (T\,X \to T\,Y)$
- Every adjunction gives rise to a monad where
  - $T = G \,.\, F$
  - $\eta = \eta$
  - $(-)^* = G \,.\, \Phi^{-1}$

Key idea:
A monad T
relative to a
functor J

Relative Monad example (Untyped lambda terms)

```
data Lam : Nat → Set where
  var : Fin n → Lam n
  lam : Lam (suc n) → Lam n
  app : Lam n → Lam n → Lam n

T = Lam, J = Fin, η = var, and
(-)* : (Fin m → Lam n) → Lam m → Lam n
```

Provides a monadic interface for substitution

Relative Monad example 2 (Vector spaces)

```
F : Nat → Set
F n = Fin n → Nat -- any semiring would do

where T = F and J = Fin and:

η : ∀{n} → Fin n → F n
η a = λ b → if a == b then 1 else 0

bind : ∀{m n} → (Fin m → F n) → F m → F n
bind f v = λ b → Σ m (λ a → v a * f a b)
```

Results

- Arrows are an instance of relative monads where J is the Yoneda embedding
- Relative monads are lax monoidal objects in a lax monoidal category (with some extra conditions on J)
- With extra extra conditions the left Kan extension of a relative monad is an ordinary monad

Self advertisement

- Paper "Monads need not be endofunctors" should/must be ready in a few days.
- Short paper "Machine assisted proofs in the theory of monads" accepted for NWPT 2009 - ongoing formalisation of monads and associated theory in Agda
- Both available at http://cs.ioc.ee/~james
- Both joint work with Tarmo and Thorsten

“All concepts are Kan extensions”

- Saunders Mac Lane

Emergency slides

R. Kleisli category

- For a R. Monad (T, J, η, (-)*) the Kleisli cat. $\mathbf{C}_T$
  - Objects : Objects of **J**
  - Morphisms: Morphisms in **C** of type $J\,X \rightarrow T\,Y$
  - Identity: $id_T = \eta$
  - composition for any $f : J\,Y \rightarrow TZ$ and $g : J\,X \rightarrow T\,Y$ $f \,._T\, g = f^* \,.\, g$

Same as for monads just delete the Js and replace **J** *with* **C**

R. Kleisli Adjunction

- Left adjoint
  - on objects: $F_T\ X = X$
  - on morphisms: $F_T\ f = \eta \circ J\ f$
- Right adjoint
  - on objects: $G_T\ X = T\ X$
  - on objects: $G_T\ k = k^*$

Same as for monads just delete the J

R. EM Category

- For a R. Monad (T, J, η, (-)*) the Kleisli cat. $\mathbf{C}^T$
  - Objects are algebras
    - (A , a : (J X → A) → (T X→ A))
  - Morphisms are algebra morphisms f : (A,a) → (B,b)

Equivalent to ordinary presentation in the case of ordinary monads by deleting the J

R. EM Adjunction

- Left adjoint
  - on objects : $L^T Y = (T Y , \lambda f . f^*)$
  - on morphisms: $L^T f = T f$
- Right adjoint
  - on objects: $R^T (A, a) = A$
  - on morphisms: $R^T h = h$

Equivalent to ordinary presentation in the case of ordinary monads